

EE 3220 – System-on-Chip Design – 2024/25 Spring

Assignment 1 – Due date: Feb 24, 2025 (Monday) – 23:59pm

Submission method – Canvas online assignment collection box

Name: _____

Date: _____

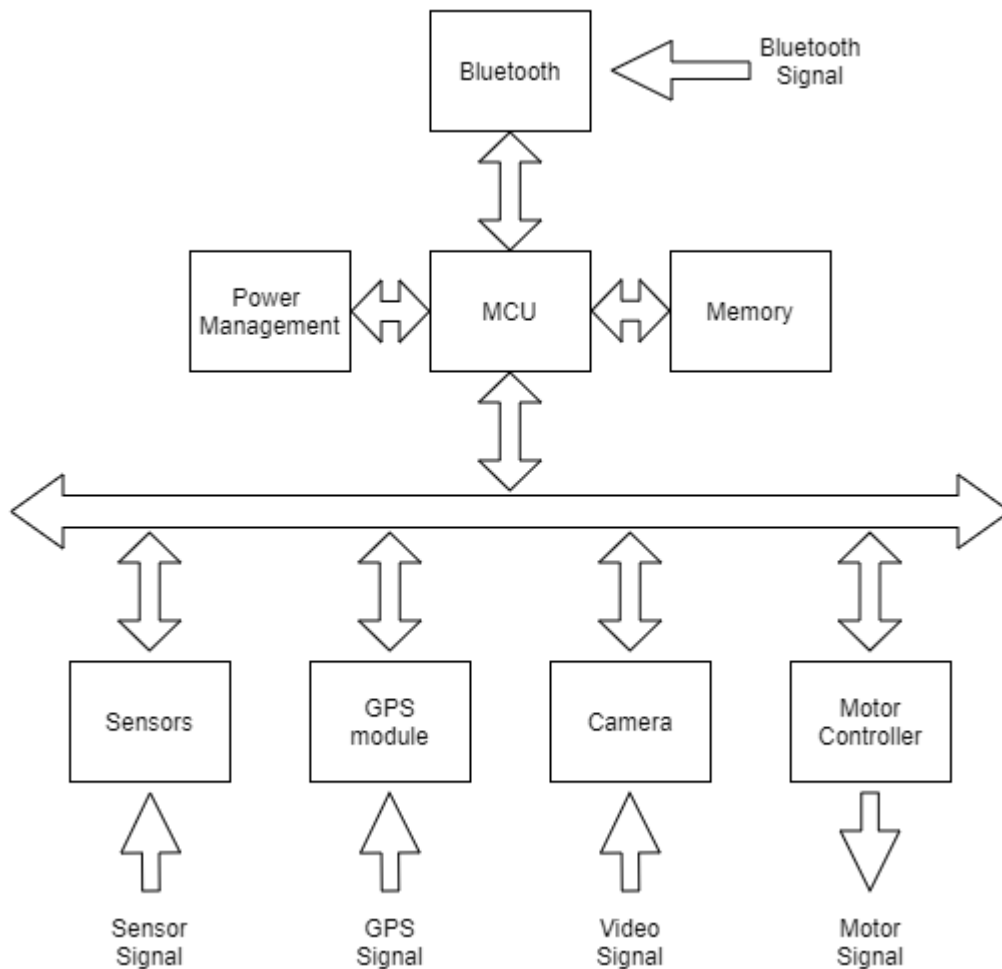
Student ID: _____

Mark: _____

Submission guidelines:

- Please prepare your assignment in “PDF” format. (1%)
- Please rename your file to “EE3220 – assignment 1 – studentID.pdf”. (1%)
- For late submission, 10% deduction per day.

Q1: Consider a controller for a drone (as shown below) as an embedded computer system.



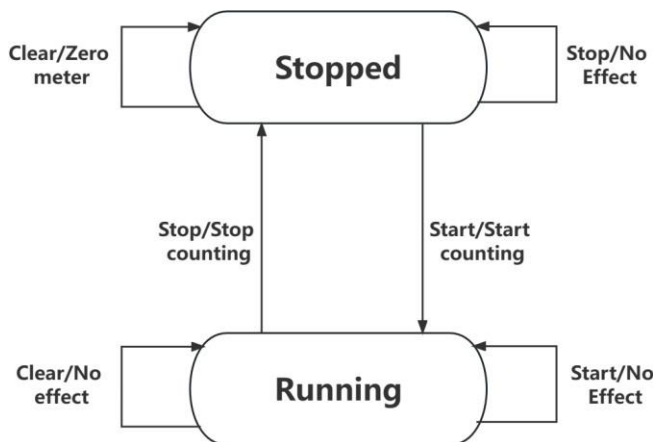
- [2 marks] What are the inputs?
- [2 marks] What are the outputs?
- [2 marks] Describe how you can control the drone's height. You can

designate the requested type of sensors.

- D. [4 marks] Describe a useful feature which could you add in software without requiring additional hardware.
- E. [4 marks] Describe a useful feature which could you add in software if the drone can be controlled by apps on mobile phones.

- A. The inputs could include Bluetooth signal, Sensor signal, GPS signal, Video signal, and GPS signal.
 - B. The output could include Motor signal.
 - C. One can use the height sensor (like a laser sensor or ultrasonic sensor) to get the height information and control the monitor signal based on the sensor signal.
- For question D and E, you can design by yourself with sufficient instructions.

For Q2-Q4, consider a depth measurement watch with the recording of work time as described here. The state machine below presents the desired behavior.



It has the following hardware.

- Buttons for Start, Stop, Depth and Clear functions.
 - Pressing **Start** starts the stopwatch running. If pressed multiple times, stopwatch continues running without resetting elapsed time.
 - Pressing **Stop** stops the stopwatch from counting.
 - Pressing **Clear** zeroes out the elapsed time if the stopwatch is not running. If it is running, the clear button is ignored.
 - Pressing **Time** records the current time. If pressed multiple times, different time data are recorded to show the working time. If it is at the stop state, takes no effect.
 - The depth of current time will be displayed on the LCD screen in real time.
- A timer which triggers an interrupt every 1 ms. The timer drives a counter which counts milliseconds since system start-up, and can be read as `elapsed_time_counter`.

- A display to show elapsed time with 1 ms resolution. The display must be updated 20 times per second.
- A “Time” button, which records the time for the working time as shown in the above Apple Watch (as an example).

Q2: Design pseudocode for the software using **event-triggered scheduling with interrupts**. Please add certain comments and instructions to introduce your design and functions. Assume that each button can generate an interrupt. [10 marks]

- Use a variable called **state** to indicate whether the stopwatch is stopped or running or recording
- Use a variable called **elapsed_time** to track how much time has elapsed since the start button was pressed.
- Use a variable called **display_delay** to track how many milliseconds remain until the display needs to be updated again.
- Use a variable called **work_time** to record the current work time.

The sample solution of Q2 – Q4 shows the basic idea of pseudocode, you can finish in your own ways with instructions.

Main thread:

```
state =
stopped
display_delay = 50
elapsed_time = 0
work_time = 0
```

Start ISR:

```
state = running
```

Timer ISR:

```
if state == running
elapsed_time += 1
ms work_time +=
1ms display_delay -
= 1
if display_delay == 0
{ display_delay = 50
display
elapsed_time
}
```

Stop ISR:

```
state = stopped
```

Clear ISR

```
if state == stopped  
    elapsed_time = 0
```

Q3: Now design pseudocode for the software using a **static scheduler without using any interrupts**. Assume that the timer updates a hardware register called **elapsed_time_register** every millisecond. [10 marks]

- Use a variable called **state** to indicate whether the stopwatch is stopped or running
- Use a variable called **start_time** to record when the start button was pressed.
- Use a variable called **stop_time** to record when the stop button was pressed.
- Use a variable called **work_time** to record when the time button was pressed.
- Use a variable called **next_display_update** to indicate when the display needs to be updated next.

```
state = stopped  
display elapsed_time_counter  
next_display_update =  
elapsed_time_counter + 50 while (1) {  
    if start switch pressed {  
        if state == stopped {  
            start_time = elapsed_time_counter  
            state = running  
        }  
    }  
  
    if stop switch pressed {  
        if state == running {  
            stop_time =  
            elapsed_time_counter state =  
            stopped  
        }  
    }  
  
    if clear switch  
    pressed { if state  
    == stopped {  
        start_time = stop_time  
    }  
}  
  
    if time switch pressed {
```

```

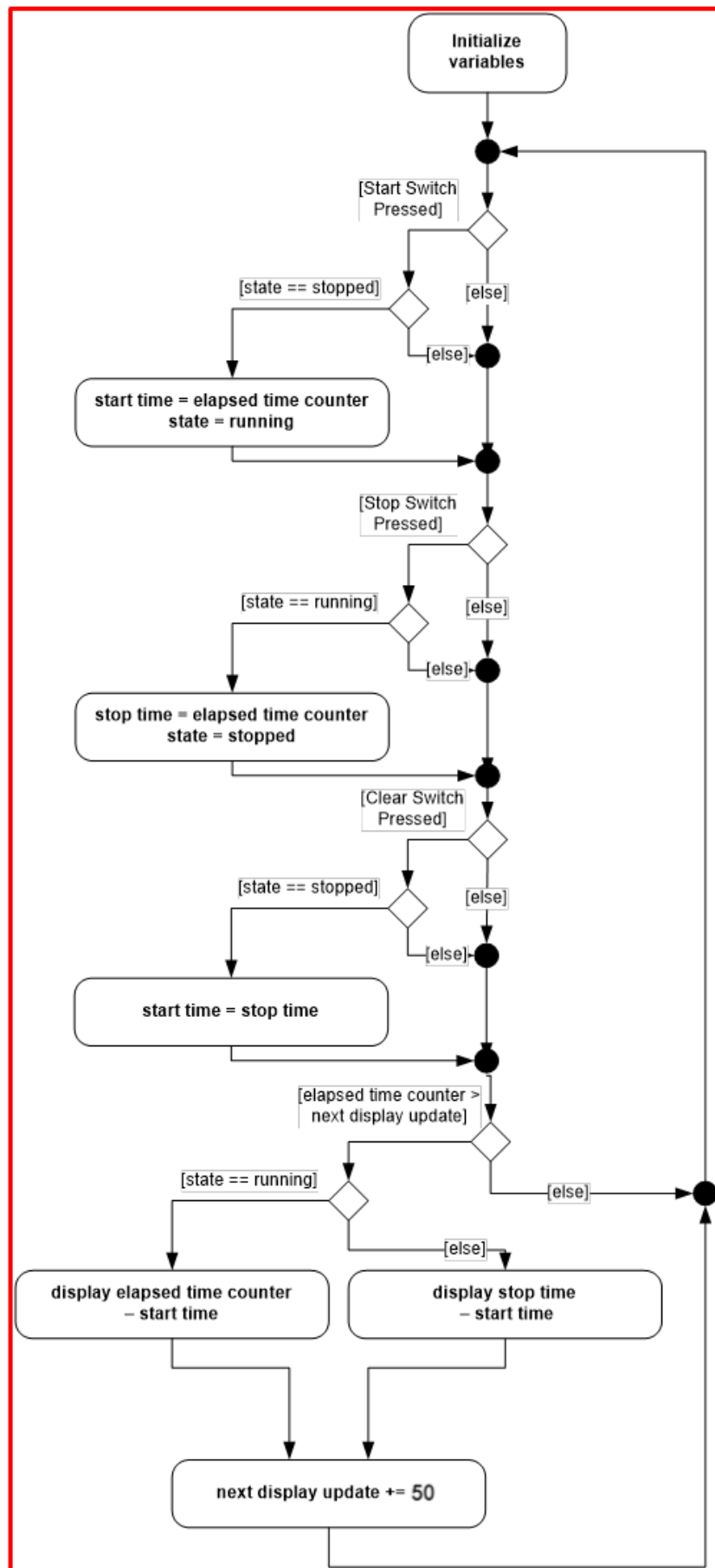
        if state == running {
            display
            work_time
            work_period
            +=1
        }
    }

    if elapsed_time_counter >
        next_display_update { if (state ==
        running)
            display elapsed_time_counter - start_time
        else
            display stop_time - start_time
        next_display_update = next_display_update + 50
    }
}

```

Q4: Create a flowchart to represent your solution to the question Q3. [10 marks]

This figure shows one solution of stopwatch with basic functions. For the time button, you can also add this process at any proper level in this figure. You can also design your process based on your own ideas.



Q5: Assume a wafer size of 20 inches, a die size of 3.0cm², defect is 1.5 / cm². Alpha is 3. Please use the equations provided in the lecture notes, determine the die yield of this CMOS process. [4 marks]

$$\text{die yield} = \left(1 + \frac{\text{defects per unit area} \times \text{die area}}{\alpha} \right)^{-\alpha}$$

and get die yield is 0.064

[2 marks for the formula, 2 marks for result]

Q6: A modern desktop processor may contain 1 billion transistors in a chip area of 160 mm². If Moore's Law continues to apply, what would be the chip area for those 1 billion transistors after 10 years (assume the cycle is 24 months)? What percentage is that area of the original area? [6 marks]

After 10 years, followings Moore's Law, there will develop for 5 cycles.

[2 marks]

So, $160/32 = 5 \text{ mm}^2$ and $5/160 = 3.125\%$ [4 marks]

Q7: Consider a system with the following tasks. We wish to **minimize** the response time for task E. For each type of scheduler, describe the sequence of processing activities which will lead to the minimum and the maximum response times for task E. Assume that each task is ready to run and there are no further task releases. [12 marks]

Task	Duration
A	3
B	2
C	8
D	1
E	5

- A. Static, non-preemptive scheduler
- B. Dynamic, non-preemptive scheduler
- C. Dynamic, preemptive scheduler

For this question, you do not need to focus on certain algorithms, just consider the best cases and the worst cases [2 marks each case]:

- a. Best case: Task E starts at time 0. $Tr = 0 + 5 = 5$
Worst case: Task E starts at last after ABCD, $Tr = 0 + 3 + 2 + 8 + 1 + 5 = 19$
- b. Best Case: Task E starts immediately (at time 0).
Worst Case: Longest task (C) just started running ϵ time units ago. E won't run until C finishes.
 $Tr = 0 + 8 - \epsilon + 5 = 13 - \epsilon$
- c. Best Case: Task E starts immediately (at time 0).
Worst Case: Longest task (C) just started running ϵ time units ago, but it is preempted by E.
 $Tr = 0 + 5 = 5$

Q8: An embedded operating system uses the Shortest Job First scheduling algorithm. Consider the arrival times and execution times for the following processes:

Process	Burst time	Arrival time
P1	20	0
P2	25	10
P3	10	30
P4	15	50

Please draw the figure of all the process with burst time and wait time. [12 marks]

The process is as follows. [8 marks]

0	20	20	45	45	55	55	70
P1		P2		P3		P4	

Wait time of P1 is 0, P2 is $20 - 10 = 10$, P3 is $45 - 30 = 15$, P4 is $55 - 50 = 5$
Ave: $(0 + 10 + 15 + 5) / 4 = 7.5$ [4 marks]

Q9: Consider the following table of arrival time and burst time for four processes P0, P1, P2 and P3.

Process	Burst time	Arrival time
P1	7 ms	0 ms
P2	4 ms	1 ms
P3	5 ms	2 ms
P4	3 ms	4 ms

Now, the CPU time is known, please compare each conventional scheduling algorithm, and select one with shortest wait time. [12 marks]

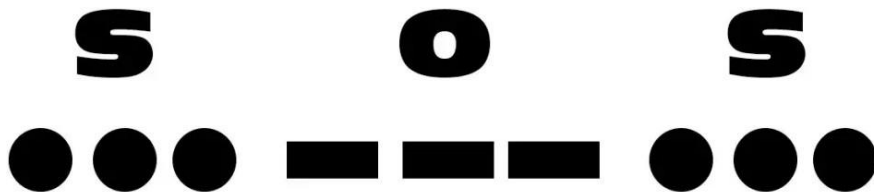
The algorithm with shortest wait time is the shortest remaining time. The process figure is shown below. [8 marks]

0	1	1	5	5	8	8	13	13	19
P1		P2		P4		P3		P1	

Wait time of P1 is $19 - 7 = 12$, P2 is $5 - 1 - 4 = 0$, P3 is $13 - 2 - 5 = 6$,
P4 is $8 - 4 - 3 = 1$
Ave: $(0 + 12 + 6 + 1) / 4 = 4.75$ [4 marks]

Q10: Please answer this question based on the content in tutorial 3 – ARM Blinky Project.

If you're trapped as a hostage and you're unable to speak, you can try to build an embedded system to start blinking for help! This clever SOS signal can be done by blinking fast three times, blinking slowly three times, and then blinking fast three times once more. Please design the program on ARM and realize the blinking of SOS.



Here is an example program: [10 marks]

```
#include "mbed.h"
// Blinking rate in milliseconds
#define BLINKING_RATE_S    300ms
#define BLINKING_RATE_L    1000ms

// main() runs in its own thread in the OS
int main() {
    // Initialise pin LED1 as
    output DigitalOut
    led(LED1);

    while (true) {
        for (int i = 0; i<6;i++) { //
            3 times led = !led;
            ThisThread::sleep_for(BLINKING_RATE_S);
        }
        for (int i = 0;
            i<6;i++) { led
            = !led;
            ThisThread::sleep_for(BLINKING_RATE_L);
        }
        for (int i = 0;
            i<6;i++) { led
            = !led;
            ThisThread::sleep_for(BLINKING_RATE_S);  }}}}
```

~ END ~